

Welcome to exceed5



Seagate



# Agreement

- ▶ โปรดรอฟังข่าวว่าพรุ่งนี้จะมาเรียนกี่โมง :D
- ▶ คำนี้ไม่เน้นความรู้ เน้นความเข้าใจ เน้นปฏิบัติเอง
  - ดังนั้นพี่จะสอนน้อยๆ เป็นเรื่องปกติ
- ▶ งานที่มีส่วนใหญ่มักมีหลายระดับ
  - โปรดใช้ความพยายามในการทำงานให้เต็มที่ ทำให้ไปไกลที่สุดเท่าที่จะทำได้

## »» Review about C# programming

# Namespace & Class

```
namespace Engineering {  
    namespace Computer {  
        class CPE {  
            int age;  
            bool AreYouKrean;  
            void TopKreanTak () {  
            }  
        }  
        class SKE {  
        }  
    }  
    class Chemical {  
    }  
}
```

# Namespace & Class

- ▶ Class คือกลุ่มของตัวแปรและเมธอด
- ▶ Namespace คือกลุ่มของ Class หรือ Namespace
  - จะประกาศตัวแปรหรือเมธอดใน Namespace เลยไม่ได้

# Using

```
System.Console.WriteLine("Exceed");
```

```
using System;
```

```
Console.WriteLine("Exceed");
```

```
using System.Console;
```

```
WriteLine("Exceed");
```



- ▶ C# มีเมธอดพื้นฐานให้เราใช้อยู่แล้ว
- ▶ การใช้งานเมธอด ต้องเรียกผ่านชื่อของ Namespace เข้าไปที่ละชั้น
  - การใช้ using จะย่อการเรียกชื่อ Namespace ได้
  - ตัวอย่างล่างสุดผิด เพราะ Console ไม่ใช่ชื่อ Namespace จะใช้ using ช่วยย่อไม่ได้

# Data and Variable Type

- ▶ int
- ▶ float
- ▶ double
- ▶ char
- ▶ string
- ▶ bool (only true, false)

```
int a = 3;
```

```
char b = '3';
```

```
string b = "3";
```

# Type Conversion #1

## ► Automatic

- การทำงานบางอย่าง ประเภทจะถูกแปลงโดยอัตโนมัติ
- การใช้งานบางเมรอด ประเภทจะถูกแปลงโดยอัตโนมัติได้เช่นกัน
- จะเกิดขึ้นได้เฉพาะกรณี ขนาดข้อมูลเล็กกว่าขนาดที่ต้องการเท่านั้น

```
double a = 2 + 3;
```

```
string b = "cpe" + 21;
```

```
double c = Math.Pow(2,3) ;
```

```
int d = 2.5 + 3
```





# Type Conversion #2

## ► Use defined method

- `xxx.Parse()`

- เช่น `Int32.Parse(data)` คำนวณข้อมูล `data` (string) ที่เป็นข้อมูลแบบ integer
- สำหรับประเภท Integer ใช้ `int.Parse(data)` ก็ได้

- `Convert.Toxxx()`

- เช่น `Convert.ToString(data)` คำนวณข้อมูล `data` ที่เป็นข้อมูลแบบ string

- สำหรับตัวแปรแบบจะทุกประเภทเราสามารถเรียกเมธอด `.ToString()` ต่อท้าย เพื่อเรียกข้อมูลเป็นประเภท string ได้

# Type Conversion #3

## ► Casting

- ใช้ในการบังคับให้ข้อมูลต่างๆ มีประเภทตามที่เรากำหนด
- โดยการใส่ (*newdatatype*) ปะหน้าไว้
- ทำได้ในบางรูปแบบเท่านั้น (ทดลองใช้เองว่าแบบไหนทำได้บ้าง)
- โดยปกติเรามักจะใช้ในกรณีที่ไม่มีเมธอดแปลงให้เช่น กรณีที่เราไม่รู้ประเภทข้อมูลต้นฉบับ หรือการแปลงจาก Enum เป็น Integer

```
int c = (int)2.1 + 3;           // c = 5
```

```
int d = (int)Math.Round(6.5); // d = 6
```

## »» Control Structure

# Condition sentence

- ▶ คือประโยคเงื่อนไขที่คืนค่าเป็น boolean (true หรือ false)
- ▶ สัญลักษณ์ที่ใช้ในการสร้างเงื่อนไข
  - == เท่ากันหรือไม่ (ต้องเท่ากับสองอันเท่านั้น ถ้าอันเดียวจะเป็น expression)
  - >, >=, <, <=
- ▶ การเชื่อมประโยคเงื่อนไขด้วยกันทางตรรกศาสตร์
  - && และ
  - || หรือ
  - ! นิเสธ

# IF

```
if (condition1) {  
    //do if condition1 is true  
}
```

```
else if (condition2) {  
    //do if condition2 is true  
}
```

```
else if (condition3) {  
    //do if condition2 is true  
}
```

```
else {  
    //do if all condition above is false  
}
```

optional

# IF (Example)

```
if (score > 80) {  
    Console.WriteLine("U Get A");  
}  
else (score > 50) {  
    Console.WriteLine("U Get C");  
}  
else {  
    Console.WriteLine("U failed!");  
}
```

- ▶ ตัวอย่างโปรแกรมตัดเกรด A, C, F
- ▶ การใช้ควรคำนึงถึงลำดับในการคิดด้วย เพราะถ้าเงื่อนไขใดจริงแล้วเงื่อนไขอื่นๆ ด้านล่างของ IF ชุดนั้นจะถูกข้ามไปทันที

# Switch

```
switch (expression) {  
    case value1 :  
        // do if variable==value1  
        break;  
    case value2 :  
    case value3 :  
        // do if variable==value2  
        or ==value3  
        break;  
    default :  
        // do if variable not equal  
        to any value (optional)  
        break;  
}
```

# Switch

```
int a,b,c;  
switch (option) {  
    case "plus" :  
    case "add" : c = a+b;  
        break;  
    case "minus" : c = a-b;  
        break;  
    case "multiple" : c = a*b;  
        break;  
    case "divide" : c = a/b;  
        break;  
    default :  
        Console.WriteLine("No Action");  
}
```

- โปรแกรมเลือกบวก ลบ คูณ หารเลขสองจำนวน



# While and For

```
while (condition) {  
    // action if condition is true  
    // when end of code, check condition again  
}
```

```
// first-statement will do at the first time (and only 1 time)  
for (first-statement; condition; endloop-statement) {  
    // action if condition is true  
    // when end of code, do endloop-statement  
    then check condition again  
}
```

- ▶ เงื่อนไขจะถูกตรวจสอบ (ว่าจะทำอีกหรือไม่?) เมื่อจบรอบแต่ละครั้ง  
เท่านั้น ไม่มีการตรวจสอบระหว่างทำอยู่

# Do-While

```
do {  
    // action for 1 time, then check condition  
} while (condition);
```

- ▶ ต่างจาก While ตรงที่คำสั่งในบล็อก do จะถูกทำก่อน แล้วค่อยตรวจสอบเงื่อนไขว่าจะทำอีกหรือไม่

# While (Example)

```
int sum = 0;
string input = Console.ReadLine();
while (input != "quit") {
    sum += int.Parse(input);
    input = Console.ReadLine();
}
Console.WriteLine("Sum of all is {0}", sum);
```

- ▶ โปรแกรมหาผลบวกของจำนวนเต็มที่ผู้ใช้พิมพ์เข้ามาเรื่อย ๆ จนกว่าผู้ใช้จะพิมพ์ quit

# For (Example)

```
string input = "exceed,freshy,rubnong";  
for(int i=0; i<input.Length; i++) {  
    if (input[i] != ',') {  
        Console.Write(input[i]);  
    }  
    else {  
        Console.WriteLine();  
    }  
}
```

```
exceed  
freshy  
rubnong  
_
```

- ▶ โปรแกรมแบ่งสตริงที่คั่นด้วย colon โดยแสดงผลคำละหนึ่งบรรทัด

# Break and Continue

## ► ใช้กับ loop

- `break;` -- โปรแกรมออกจาก loop นั้นทันทีโดยไม่ทำโค้ดที่เหลือข้างล่างต่อ
- `continue;` -- จะกลับไปทำคำสั่งส่วน `endloop-statement` (ถ้ามี) แล้วตรวจเงื่อนไขใหม่ทันทีโดยไม่ทำโค้ดที่เหลือข้างล่างต่อ

```
string grade = "ACBDFWDAAA";  
for (int i=0; i<grade.Length; i++) {  
    if (grade[i] == 'F') {  
        // break;  
        // continue;  
    }  
    Console.Write(grade[i]);  
}
```

ACBD\_

ACBDWDAAA\_

## >> Array

# Array

- ▶ กลุ่มของตัวแปรที่ใช้หมายเลขช่องในการระบุค่าแต่ละตัว
- ▶ ตัวอย่างการประกาศอาเรย์

```
char [] exceed;  
exceed = new char [10];
```

```
char [] exceed = new char [10];
```

- ▶ ตัวอย่างการประกาศอาเรย์โดยการกำหนดค่าเริ่มต้น

```
int [] exceed = new char [] { 'a' , 'b' , 'c' , 'd' , 'e' };
```

exceed	0	1	2	3	4
	a	b	c	d	e

# Array #2

## ▶ ตัวอย่างการพูดถึงค่าในอาเรย์

```
int [] exceed = new char [] { 'a', 'b', 'c', 'd', 'e' };  
Console.WriteLine(exceed[0]); // output is a  
Console.WriteLine(exceed[1]); // output is b  
Console.WriteLine(exceed[5]); // runtime error!
```

## ▶ ตัวอย่างโปรแกรมนับจำนวนคู่ในอาเรย์

```
int [] exceed = new int [] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
int even = 0;  
for(int i=0; i<10; i++) {  
    if(exceed[i] % 2 == 0) even++;  
}  
Console.WriteLine("Have {0} even num.", even);
```

Have 5 even num.



# Array #3

## ► ขนาดของ Array

- *ArrayName.Length* เป็น property ที่ให้ค่า “จำนวนช่อง” ของ Array นั้น
- *ArrayName.GetLength(i)* เป็น method ที่คืนค่า “ขนาด” ของ Array ในมิติที่ i
  - สำหรับ 2 มิติ; *GetLength(0)* ได้จำนวนแถว, *GetLength(1)* ได้จำนวนหลัก

```
int [,] exceed = new int [,] { {1,2,3}, {4,5,6} }  
Console.WriteLine(exceed.Length);  
// output is 6  
Console.WriteLine(exceed.GetLength(0));  
// output is 3  
Console.WriteLine(exceed.GetLength(1));  
// output is 2
```

## >> Method

# Method Creation

## ► โครงสร้าง Method

### ○ Header

```
int plus ( int x , int y )
```

Return type

Method's name

Parameter(s)

### ○ Body

```
{  
    int sum = x + y;  
    return sum;  
}
```

# Using method

## ▶ Calling Method

- `ClassName.MethodName(Parameter);`
- ถ้าเรียกใช้ใน Class เดียวกัน ไม่ต้องใส่ `ClassName` ก็ได้

```
int z = plus ( x , y );
```

## ▶ Advantage of using method

- Divide and Conquer
- Reuse
- Easier to understand

# Method Overloading

- ▶ เราสามารถประกาศเมธอดชื่อซ้ำกัน แต่อาร์กิวเมนต์ต่างกันได้
- ▶ เวลาใช้โปรแกรมจะตรวจสอบว่าอาร์กิวเมนต์ตรงกับรูปแบบไหน ก็จะทำตามรูปแบบนั้น

```
int sum (int x, int y) { return x+y; }  
int sum (int x, int y, int z) {  
    return x+y+z;  
}  
static void Main () {  
    Console.WriteLine(sum(1,2));    // 3  
    Console.WriteLine(sum(1,2,3));  // 6  
}
```

# Scope of Variables

ขอบเขตของตัวแปรที่ตัวแปรสามารถถูกเรียกใช้ได้นั้น จะอยู่ตั้งแต่ประกาศตัวแปรจนถึงสิ้นสุด Block นั้นๆ (จนถึง } )

## ▶ Local Variable

- ตัวแปรที่ประกาศใน Method หรือ Block
- เรียกใช้ได้เฉพาะภายใน Method หรือ Block ที่ถูกประกาศเท่านั้น

## ▶ Global Variable

- ตัวแปรที่ประกาศใน Class และอยู่นอก Method
- เรียกใช้ได้จากทุกๆ Method ใน Class นั้น

```
class Test
{
    int a = 5;
    static void Main ()
    {
        int b = 55;
        if(b%5==0)
        {
            int c = add(b) ;
            Console.WriteLine(c) ;
        }
    }
    static int add ( int x )
    {
        int sum = x + a;
        return sum;
    }
}
```

»» More Technique



# Enum #1

- ▶ เป็นการกำหนดประเภทข้อมูลใหม่ ที่เราสามารถกำหนดสมาชิกแต่ละตัวเป็นค่าคงที่โดยใช้คำเป็นตัวแทนได้
- ▶ ประกาศรูปแบบใน Namespace หรือ Class
  - ประกาศรูปแบบใน Method ไม่ได้
- ▶ ตัวอย่างการประกาศ

```
enum UserType {  
    Guest=0, Member=1, Admin=2  
}
```

- ▶ หากไม่สนใจค่าที่เป็นตัวเลขสามารถละการกำหนดค่าตัวเลขให้ข้อมูลแต่ละตัวได้ โดยจะถูกกำหนดให้ตัวแรกเป็น 0, 1, 2, ... โดยอัตโนมัติ

# Enum #2

## ► ตัวอย่างการใช้งาน

```
UserType a = ...;  
if (a == UserType.Guest) {  
    ShowLoginBox();  
}
```

- สังเกตว่าผู้เขียนโปรแกรมไม่จำเป็นต้องจำว่า Guest คือเลขอะไร ทำให้ไม่สับสนในการเขียน Code กับคนอื่น หรือการกลับมาอ่าน Code วนหลัง

# Enum #3

- ▶ หาก Enum อยู่ใน Class อื่น หรือ Namespace อื่นที่ไม่ใช่ Class, Namespace ที่ทำงานอยู่ ต้องระบุตำแหน่งให้ชัดเจนด้วย

```
namespace User {  
    enum UserType { Guest, Member, Admin }  
}  
  
namespace Page {  
    class adminPage {  
        bool checkAdmin (user a) {  
            if (a == User.UserType.Guest)  
                return true;  
        }  
    }  
}  
}
```

# Try-Catch

- ▶ เป็น block สำหรับดักจับความผิดพลาดที่ทำให้โปรแกรมหยุดทำงานได้

```
try { // something we want to try }  
catch { // do this if an error occurs  
        when doing something in try{} }
```

- ▶ มักใช้ในกรณีที่เราไม่อาจควบคุมได้ว่าจะเกิด Error หรือไม่ และไม่อยากจะให้เกิด Error ขึ้น

# Try-Catch (Example)

- ▶ ตัวอย่างโปรแกรมหารเลขที่ป้องกัน Run Time Error แล้ว

```
double a, b;  
a = ...; b = ...;  
try {  
    Console.WriteLine("a / b is {0}",a/b);  
}  
catch {  
    Console.WriteLine("Error with dividing!");  
}
```






# Task : Exceed Expression

- ▶ ให้เขียนโปรแกรมคำนวณคำสั่งคณิตศาสตร์โดยมีข้อกำหนดดังนี้
  - รับคำสั่งเป็น string ที่ประกอบด้วยตัวเลข (ได้ทั้งจำนวนเต็มและทศนิยม) และเครื่องหมาย  $+$   $-$   $/$   $*$
  - ให้คำนวณคำสั่งที่ได้ จากเครื่องหมายขวาสุดไปซ้ายสุด แล้วแสดงคำตอบ
    - เช่น  $2+3$  ได้ผลลัพธ์เป็น 5
    - $2+6*7/15+18$  ได้ผลลัพธ์เป็น 3.2726
  - เมื่อแสดงคำตอบแล้วให้รอรับคำสั่งใหม่ไปเรื่อยๆ จนกว่าผู้ใช้จะพิมพ์ quit
- ▶ รายละเอียดอ่านได้ที่

[http://exceed.cpe.ku.ac.th/wiki/Task:Exceed\\_Expression](http://exceed.cpe.ku.ac.th/wiki/Task:Exceed_Expression)

»» Windows Apps.

# Icon in AutoComplete of VC#

- ▶ Namespace  System namespace System
- ▶ Object (สิ่งที่ถูกสร้างมาจาก Class)  Form1 class WindowsFormsApplication1.Form1
- ▶ Method  Write void Console.Write(string value)
- ▶ Property  Text string Form.Text
- ▶ Event  Click EventHandler Control.Click
- ▶ มีอย่างอื่นอีก ลองสังเกตดูนะ



# Task : The KAK Challenge

- ▶ สร้างเกม The KAK Challenge
- ▶ รายละเอียดอ่านได้ที่

[http://exceed.cpe.ku.ac.th/wiki/Task:The\\_KAK\\_Challenge](http://exceed.cpe.ku.ac.th/wiki/Task:The_KAK_Challenge)

# Task : Exceed Calculator

- ▶ สร้างโปรแกรม Exceed Calculator
- ▶ รายละเอียดอ่านได้ที่

[http://exceed.cpe.ku.ac.th/wiki/Task:Exceed\\_Calculator](http://exceed.cpe.ku.ac.th/wiki/Task:Exceed_Calculator)